* Q5.1 – Assume that a 10-D base cuboid contains only three base cells: (1) (*a1, d2, d3, d4, …, d9, d10),* (2) (*d1, b2, d3, d4, …, d9, d10*), and (3) (*d1, d2, c3, d4, …, d9, d10*), where a1 ≠ d1, b2 ≠ d2, and c3 ≠ d3. The measure of the cube is count().
  + (a) How many *nonempty* cuboids will a full data cube contain?

210 = 1,024 *nonempty* cuboids would be contained in a full data cube

* + (b) How many *nonempty* aggregate (i.e., nonbase) cells will a full cube contain?
    - Each cell generates 210-1 = 1,023 nonempty aggregate cells with overlaps removed
    - Cells that overlap once is 3 \* 27 = 384
    - Cells that overlap twice is 1 \* 27 = 128
    - Total nonempty aggregate cells = 3\*(1023) – 384 – 2\*(128) = 2,429

2,429 nonempty aggregate cells

* + (c) How many *nonempty* aggregate cells will an iceberg cube contain if the condition of the iceberg cube is “count ≥ 2”?
    - (\*,\*,\*,*d4, …, d9, d10)* has a count of 3
    - (\*,\*,*d3, d4, …, d9, d10*) has a count 2
    - (\*,*d2,\*, d4, …, d9, d10*) has a count 2
    - (*d1,\*,\*, d4, …, d9, d10*) has a count 2
    - 4 \* 27 = 512

512 nonempty aggregate cells in the iceberg cube

* + (d) A cell, *c*, is a *closed cell* if there exists no cell, *d*, such that *d* is a specialization of cell *c* (i.e., *d* is obtained by replacing a \* in *c* by a non-\* value) and *d* has the same measure value *c*. A *closed cube* is a data cube consisting of only closed cells. How many closed cells are in the full cube?
    - 7 closed cells total:
    - (*a1, d2, d3, d4, …, d9, d10)*
    - (*d1, b2, d3, d4, …, d9, d10*)
    - (*d1, d2, c3, d4, …, d9, d10*)
    - (\*,\*,\*,*d4, …, d9, d10)*
    - (\*,\*,*d3, d4, …, d9, d10*)
    - (\*,*d2,\*, d4, …, d9, d10*)
    - (*d1,\*,\*, d4, …, d9, d10*)
* Q5.2 – There are several typical cube computation methods, such as *MultiWay* [ZDN97], *BUC* [BR99], and *Star-Cubing* [XHLW03]. Briefly describe these three methods (i.e., use one or two lines to outline the key points), and compare their feasibility and performance under the following conditions:
  + - Multiway Computation Method Key Points:
      * Utilizes a “bottom-up” algorithm with multi-dimensional chunks
      * Performs simultaneous aggregation on arrays
      * Intermediate aggregation values are used to compute ancestor
    - BUC Computation Method Key Points:
      * Utilizes “top-down” algorithm
      * Utilizes dimension partitioning for iceberg pruning
      * There are no simultaneous aggregations being performed
    - Star-Cubing Computation Method Key Points:
      * Utilizes both top-down and bottom-up algorithms
        + Bottom-up is a sub-layer to allow Apriori pruning
      * Allows the exploration of shared dimensions and shared computations
  + (a) Computing a dense full cube of low dimensionality (e.g., less than eight dimensions)
    - In this situation, all the methods would be acceptable; however, MultiWay and Star-Cubing would probably be the better choice.
  + (b) Computing an iceberg cube of around 10 dimensions with a highly skewed data distribution.
    - In this situation, MultiWay or Star-Cubing would work for creating an iceberg cube; however, BUC would not be a good choice for highly skewed data.
  + (c) Computing a sparse iceberg cube of high dimensionality (e.g., over 100 dimensions).
    - In this situation, MultiWay would be the best choice for handling an iceberg cube. Star-Cubing and BUC are not every effective with data sets with high dimensionality.
* Q5.4 – Suppose that a base cuboid has three dimensions, *A, B, C,* with the following number of cells: |A| = 1,000,000, |B| = 100, and |C| = 1,000. Suppose that each dimension is evenly partitioned into 10 portions for *chunking*.
  + (a) Assuming each dimension has only one level, draw the complete lattice of the cube.



* + (b) If each cube cell stores one measure with four bytes, what is the total size of the computed cube if the cube is *dense?*

Total size of cube by each level:

* + - All: 1
    - A: 1,000,000, B: 100, C: 1,000
    - AB: 100,000,000, BC: 100,000, and AC: 1,000,000,000,000
    - ABC: 100,000,000,000
    - Each measure is four bytes
    - Total measures (101,101,101,101) \* 4 bytes = 404,404,404,404 bytes

If the cube is dense, then the total size would be 404,404,404,404 bytes

* + (c) State the order for computing the chunks in the cube that requires the least amount of space, and compute the total amount of main memory space required for computing the 2-D planes.
    - The order for computing the chunks that requires the least amount of space would be to compute the whole *BC* plane, then one column of the *BA* plane, and then one chunk of the *CA* plane.
    - BC Plane: 100,000 cells x 4 bytes = 400,000 memory units
    - BA Column: 10,000,000 cells x 4 bytes = 40,000,000 memory units
    - CA Chunk: 10,000,000 cells x 4 bytes = 40,000,000 memory units
    - Total Memory Space Required: 80,400,000 memory units

